##### A Project report on

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

###### A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

Submitted by

**M.YASHWANTH**

(20H51A05C8)

**N. KOUSHIK**

(20H51A05G2)/

**P. ANUSHA**

(20H51A05J3)

Under the esteemed guidance of

**Mr. T. UPENDER**

**Assistant Professor**

**CSE**



**Department of Computer Science and Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

(UGC Autonomous)

\*Approved by AICTE \*Affiliated to JNTUH \*NAAC Accredited with A+ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

#### 2023- 2024

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



#### CERTIFICATE

This is to certify that the Major Project Phase I report entitled **" Creating Realistic Novel Images through Generative Neural Networks (GAN) "** being submitted by M. YASHWANTH (20H51A05C8), N. KOUSHIK (20H51A05G2), P. ANUSHA (20H51A05J3) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

###### The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

**Mr. T. Upender Dr. Siva Skandha Sanagala EXTERNAL EXAMINER**

**Assistant Professor Associate Professor and HOD**

**Dept. of CSE Dept. of CSE**

#### ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **Mr. T. Upender,** Assistant Professor, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

M.YASHWANTH 20H51A05C8

N. KOUSHIK 20H51A05G2

P. ANUSHA 20H51A05J3

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO.**

**NO.**

LIST OF FIGURES ii

ABSTRACT iii

**1** **INTRODUCTION** 1

1.1 Problem Statement 2

1.2 Research Objective 3

1.3 Project Scope and Limitations 4

**2** **BACKGROUND WORK** 7

2.1 DALL-E 2

2.1.1. Introduction 8

2.1.2. Merits, Demerits, and Challenges 8

2.1.3. Implementation of DALL-E 2 8

2.2. IMAGEN

2.2.1. Introduction 9

2.2.2. Merits, Demerits, and Challenges 9

2.2.3. Implementation of IMAGEN 9

2.3. VQ-GAN

2.3.1. Introduction 10

2.3.2. Merits, Demerits, and Challenges 10

2.3.3. Implementation of VQ-GAN 11

2.4. PARTI

2.4.1. Introduction 12

2.4.2. Merits, Demerits, and Challenges 12

2.4.3. Implementation of PARTI 12

**3 PROPOSED SYSTEM 14**

3.1. Objective of Proposed Model 15

3.2. Algorithms Used for Proposed Model 16

3.3. Designing 20

3.3.1.UML Diagram 22

3.3. Stepwise Implementation and Code 23

**4 RESULTS AND DISCUSSION 45** 4.1. Performance metrics 46

**5** **CONCLUSION** 49

5.1 Conclusion and Future Enhancement 50

**REFERENCES** **52**

**GITHUB LINK 53**

**PUBLICATION CERTIFICATES 54**

CMRCET B. Tech (CSE) Page No i

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**List of Figures**

**FIGURE NO. TITLE PAGE NO.**

2.1.3.1 DALL-E 2 Architecture 9

2.2.3.1 Imagen System Architecture 10

2.3.3.1 VQ-GAN System Architecture 11

2.4.3.1 Parti System Architecture 13

3.2.1 GAN framework 17

3.2.2 DCGAN Architecture 17

3.2.3 CapsGAN discriminator Architecture 19

3.2.4 AEGAN Architecture 19

3.2.5 GAN Discriminator 21

3.2.6 GAN Generator 22

3.2.7 GAN Discriminator and Generator 23

3.2.8 Generated fake images 26

3.2.9 GAN Framework 27

4.1.1 Graph of Performance Metrics 47

4.1.2 Generated Images by Trained Mode 48

CMRCET B. Tech (CSE) Page No ii

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

# **ABSTRACT**

The "Generative Neural Networks for Novel Image Generation" project aims to expand

creative & generative capabilities of neural networks beyond traditional discriminative models. In previous contexts, neural networks have primarily utilized for tasks involving input-to-output mappings, such as image classification and text generation. However, this project delves into the realm of generative models, where the focus shifts from making decisions to creating entirely new and unique creative content.

At its core, the project equips neural networks with the power to craft images that encapsulate the style and essence of existing training data. This synthesis of new, yet familiar, visuals introduce diversity and creativity. Beyond artistic, the project holds practical value in data augmentation, offering a solution to data scarcity by generating synthetic content that can enhance machine learning model performance.

The impact of this project extends across industries. In healthcare, it assists medical image analysis by generating realistic data for algorithm training. In fashion, it aids design by creating new patterns and styles.

The project also addresses data privacy concerns, enabling information sharing without compromising sensitive details. By forging a bridge between technology and creativity, the "Generative Neural Networks for Novel Image Generation" project innovation, enriches data science.

Moreover, the project underscores the significance of synthetic data in addressing data scarcity and privacy concerns. Synthetic data has the potential to supplement real datasets in scenarios where access to authentic data is limited or protected.

CMRCET B. Tech (CSE) Page No iii

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

# **CHAPTER 1**

**INTRODUCTION**

CMRCET B. Tech (CSE) Page No 1

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 1**

**INTRODUCTION**

* 1. **Problem Statement**

The field of neural networks has predominantly focused on discriminative models, which excel at tasks involving input-to-output mappings like image classification and text generation. While these models have achieved remarkable success in decision-making tasks, there remains a significant gap in their creative capabilities. Existing neural networks are limited in their capacity to generate entirely novel and unique content. Consequently, there is a pressing need to push the boundaries of neural network capabilities by developing generative models that can go beyond mere decision-making and instead, create innovative and original content. The "Generative Neural Networks for Novel Image Generation" project aims to address this gap by exploring and advancing the potential of generative neural networks in the realm of creative content generation.

* Limitations of traditional discriminative models: Current neural networks lack the ability to generate diverse and creative visual content, hindering data augmentation and artistic expression.
* Insufficient methods for data augmentation: The absence of effective generative models limits the potential for enhancing machine learning model performance through synthetic content generation.
* Practical applications in industries: The project addresses the need for generating new patterns, styles, and realistic data in industries such as healthcare and fashion.
* Bridging technology and creativity: The project aims to bridge the gap between technology and creativity by developing innovative approaches for generative image generation using neural networks.

"Novel" refers to something new, original, or innovative. In the context of the project description you provided, "novel images" are images that have not been seen before in the training dataset but are generated by a neural network to closely resemble the characteristics and patterns of the training images. These generated images are unique and creative.

CMRCET B. Tech (CSE) Page No 2

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

The "Generative Neural Networks for Novel Image Generation" project aims to expand the creative capabilities of neural networks beyond traditional discriminative models. In previous contexts, neural networks have primarily been utilized for tasks involving input-to-output mappings, such as image classification and text generation. However, this project delves into the realm of generative models, where the focus shifts from making decisions to creating entirely new and unique content.

The core objective of this project is to train neural networks to generate novel images that closely resemble a predefined set of training images. By learning the underlying distribution and patterns within the training data, the generative models acquire the ability to produce images that maintain the same stylistic and structural characteristics. This creative endeavor

Introduces a level of innovation distinct from conventional applications, as it involves the

synthesis of visual content rather than making determinations based on existing data.

The generative images crafted through this project possess qualities of novelty and diversity, showcasing the model's capacity to envision variations that harmonize with the original data distribution. By extending the boundaries of neural network application, this project contributes to artistic expression, creative design, and data augmentation. The novel images serve as a testament to the model's ingenuity in capturing intricate details and crafting content that captures the essence of the training data.

* 1. **Research Objective**

The objective of the "Generative Neural Networks for Novel Image Generation" project is to explore and expand the creative and generative capabilities of neural networks beyond traditional discriminative models. The project aims to equip neural networks with the ability to generate entirely new and unique creative content by synthesizing images that encapsulate the style and essence of existing training data. By delving into the realm of generative models, the project shifts the focus from decision-making tasks to the creation of creative visual content.

CMRCET B. Tech (CSE) Page No 3

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

* 1. **Project Scope and Limitations**

**Project Scope:**

**1. Expand Creative and Generative Capabilities:**

-It delves into the realm of generative models, focusing on creating entirely new and unique creative content rather than just making decisions based on existing data.

**2. Generate Novel Images:**

- The project focuses on training neural networks to generate novel images that encapsulate the style and essence of existing training data.

- These synthesized images introduce diversity and creativity, enriching the dataset with new, yet familiar, visual content.

**3. Practical Applications:**

- Beyond artistic value, the project holds practical value in various domains.

- In healthcare, it assists in medical image analysis by generating realistic data for algorithm training.

- In fashion, it aids in design by creating new patterns and styles.

- It also addresses data privacy concerns by enabling information sharing without compromising sensitive details.

**4. Data Augmentation and Data Scarcity:**

- The project offers a solution to data scarcity by generating synthetic content that can enhance machine learning model performance through data augmentation.

- Synthetic data has the potential to supplement real datasets in scenarios where access to authentic data is limited or protected.

CMRCET B. Tech (CSE) Page No 4

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**5. Impact Across Industries:**

- The impact of the project extends across various industries, including healthcare, fashion, and data science.

- It fosters innovation and creativity by bridging the gap between technology and artistic expression.

**6. Research and Development:**

- The project contributes to advancements in the field of generative modeling, leveraging technologies like Generative Adversarial Networks (GANs) to push the boundaries of creative content synthesis.

**Limitations:**

**1. Quality of Generated Images:**

- One limitation may lie in the quality of the generated images. Despite advancements in generative models, generated images may still lack the fidelity and realism of real images. The quality of generated images can be affected by factors such as dataset size, model architecture, and training parameters.

**2. Data Bias and Generalization:**

- Generative models may struggle to generalize well to unseen data or diverse datasets. If the training dataset is biased or limited in scope, the generated images may exhibit biases or lack diversity. Ensuring the diversity and representativeness of the training data is crucial for overcoming this limitation.

**3. Training Time and Resource Intensiveness:**

- Training generative models, especially complex architectures like GANs, can be computationally intensive and time-consuming.

CMRCET B. Tech (CSE) Page No 5

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**4. Mode Collapse and Training Instability:**

- GANs, in particular, are prone to mode collapse, where the generator produces limited variations of images or gets stuck generating similar samples. Training instability, such as oscillations in generator and discriminator performance, can also pose challenges in achieving stable convergence during training.

**5. Ethical and Privacy Concerns:**

- The use of generative models raises ethical considerations, particularly regarding the generation of synthetic data that resembles real individuals or sensitive information. Ensuring responsible use and mitigation of potential privacy risks is essential to address these concerns.

**6. Evaluation Metrics and Subjectivity:**

- Evaluating the quality and creativity of generated images can be subjective and challenging. While objective metrics like Frechet Inception Distance (FID) or Inception Score (IS) are commonly used, they may not always capture the perceptual quality or artistic value of generated content accurately.

**7. Domain-Specific Challenges:**

- Certain domains, such as medical imaging or fashion, may pose unique challenges for generative models. Generating realistic medical images with anatomical accuracy or capturing intricate fashion styles and textures may require specialized architectures and domain-specific expertise.

CMRCET B. Tech (CSE) Page No 6

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 2**

**BACKGROUND WORK**

CMRCET B. Tech (CSE) Page No 7

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 2**

**BACKGROUND WORK**

**2.1. DALL-E 2**

**2.1.1. Introduction**

DALL-E 2 represents a breakthrough in image and video synthesis from textual descriptions, achieved through the scaling of diffusion models. This model enables the generation of highly realistic visual content based solely on textual prompts.

**2.1.2. Merits, Demerits, and Challenges**

**- Merits:**

1. Advanced capability in generating realistic images and videos from text.

2. Potential for diverse applications in design, creativity, and content creation.

**- Demerits:**

1. Challenges may arise in scaling the model and managing computational resources.

**- Challenges:**

1. Scalability of the model for handling large datasets.

2. Optimization for efficient training and inference processes.

**2.1.3. Implementation of DALL-E 2**

- Implementation of DALL-E 2 involves leveraging state-of-the-art diffusion models and text-to-image synthesis techniques.

- Requires substantial computational infrastructure and optimization strategies for efficient operation.

- Available pre-trained models and resources for ease of implementation using deep learning libraries such as PyTorch and TensorFlow.

CMRCET B. Tech (CSE) Page No 8

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

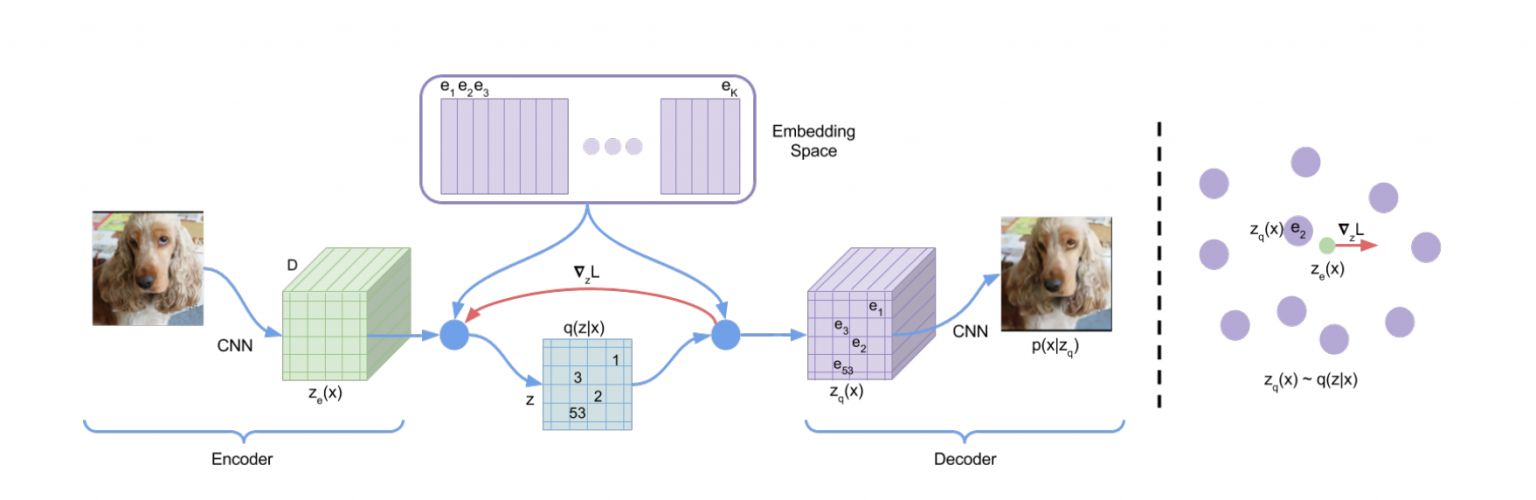


Figure 2.1.3.1: DALL-E 2 Architecture

**2.2. IMAGEN**

**2.2.1. Introduction**

Imagen stands as a pioneering diffusion model specifically tailored for text-to-image synthesis. Developed by Ramesh et al., it bridges the semantic gap between textual descriptions and visual representations, offering a novel approach to generate images from natural language prompts.

**2.2.2. Merits, Demerits, and Challenges**

**- Merits:**

1. Specialized in generating visually coherent images from textual inputs.

2. Valuable tool for creative expression and visual content generation.

**- Demerits:**

1. Challenges may arise in handling complex textual descriptions accurately.

**- Challenges:**

1. Ensuring faithful translation of semantics into visual features.

2. Addressing computational requirements for training and inference.

**2.2.3. Implementation of IMAGEN**

- Implementation of Imagen involves training diffusion models on large-scale datasets of text-image pairs.

CMRCET B. Tech (CSE) Page No 9

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

- Utilizes techniques such as masked autoencoders to capture intricate visual details.

- Requires careful tuning of hyperparameters and experimentation for optimal performance in

image generation tasks.

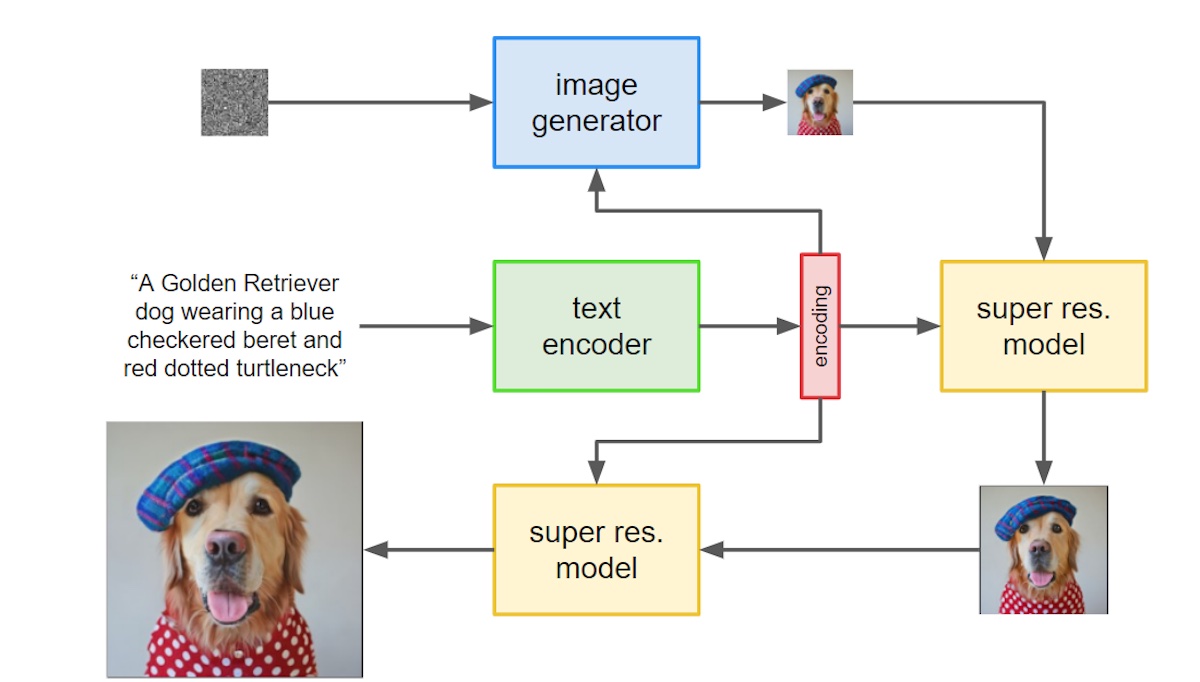


Figure 2.2.3.1: Imagen System Architecture

**2.3. VQ-GAN**

**2.3.1. Introduction**

VQ-GAN, developed by Rombach et al., is a diffusion model primarily trained on images using masked autoencoders. It focuses on capturing and synthesizing complex visual patterns through a combination of vector quantization and generative adversarial training.

**2.3.2. Merits, Demerits, and Challenges**

**- Merits:**

1. Capable of generating high-quality images with fine-grained details.

2. Versatile in various image synthesis tasks.

CMRCET B. Tech (CSE) Page No 10

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**- Demerits:**

1. May face challenges in scalability and computational efficiency during inference.

**- Challenges:**

1. Training large-scale models effectively.

2. Optimizing computational resources for efficient synthesis.

**2.3.3. Implementation of VQ-GAN**

- Implementing VQ-GAN involves training diffusion models on diverse image datasets.

- Utilizes techniques such as vector quantization to discretize continuous data representations.

- Requires efficient utilization of computational resources and optimization strategies for satisfactory results.

- Implementation of VQ-GAN involves training diffusion models on diverse image datasets, emphasizing the integration of masked autoencoders.

- The utilization of techniques such as vector quantization facilitates the discretization of continuous data representations, enhancing model performance.

- Effective implementation requires meticulous resource management and optimization strategies to achieve satisfactory results in image generation tasks.

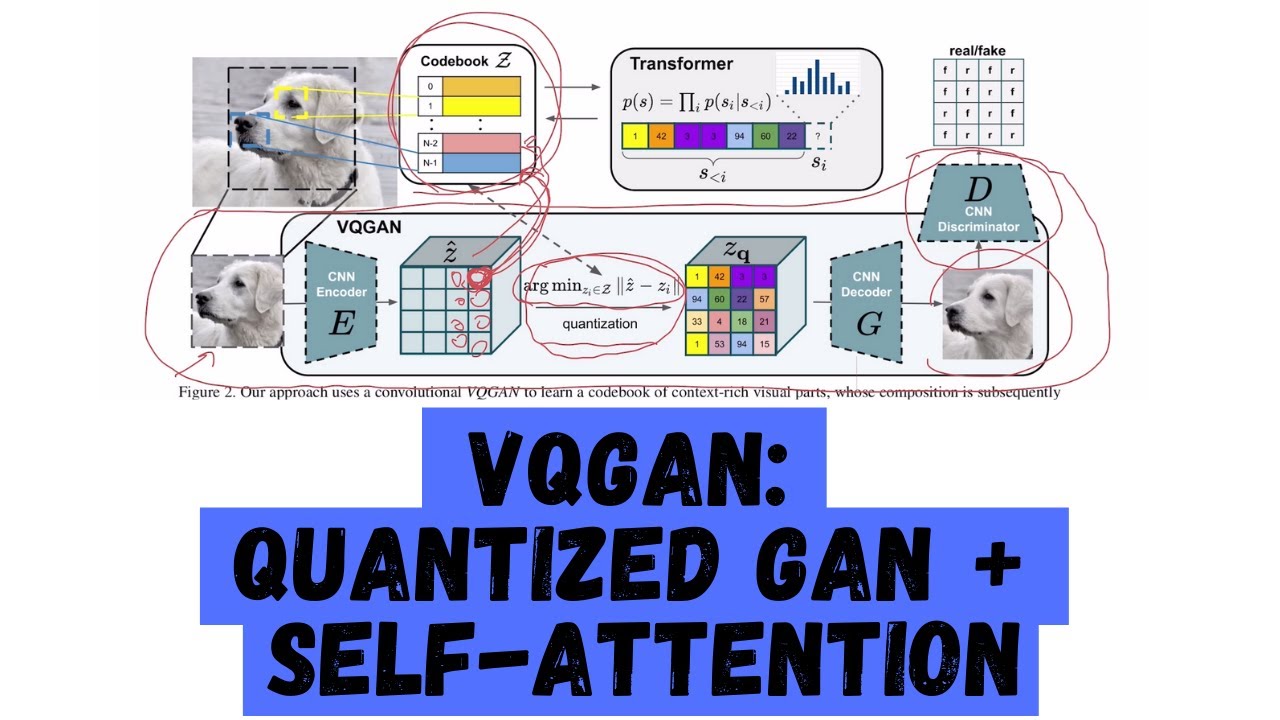


Figure 2.3.3.1: VQ-GAN System Architecture

CMRCET B. Tech (CSE) Page No 11

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**2.4. PARTI**

**2.4.1. Introduction**

Parti, introduced by Ramesh et al., is a modal-parallel diffusion model specifically designed for high-resolution image generation. It represents a significant advancement in the field by addressing the challenge of generating complex visual content with exceptional detail and fidelity.

**2.4.2. Merits, Demerits, and Challenges**

**- Merits:**

1. Specialized in generating high-resolution images with fine details.

2. Offers potential for various applications requiring detailed visual content.

**- Demerits:**

1. Potential challenges may arise in scalability and computational complexity.

**- Challenges:**

1. Optimizing the model architecture for efficient training and inference.

2. Managing computational resources effectively to handle high-resolution image synthesis.

**2.4.3. Implementation of PARTI**

- Implementation of Parti involves leveraging parallel diffusion models tailored for high-resolution image generation.

- Requires careful consideration of model architecture and optimization techniques to ensure efficient training and inference processes.

- Pre-trained models and resources may be available to facilitate implementation using popular deep learning libraries like PyTorch and TensorFlow.

CMRCET B. Tech (CSE) Page No 12

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

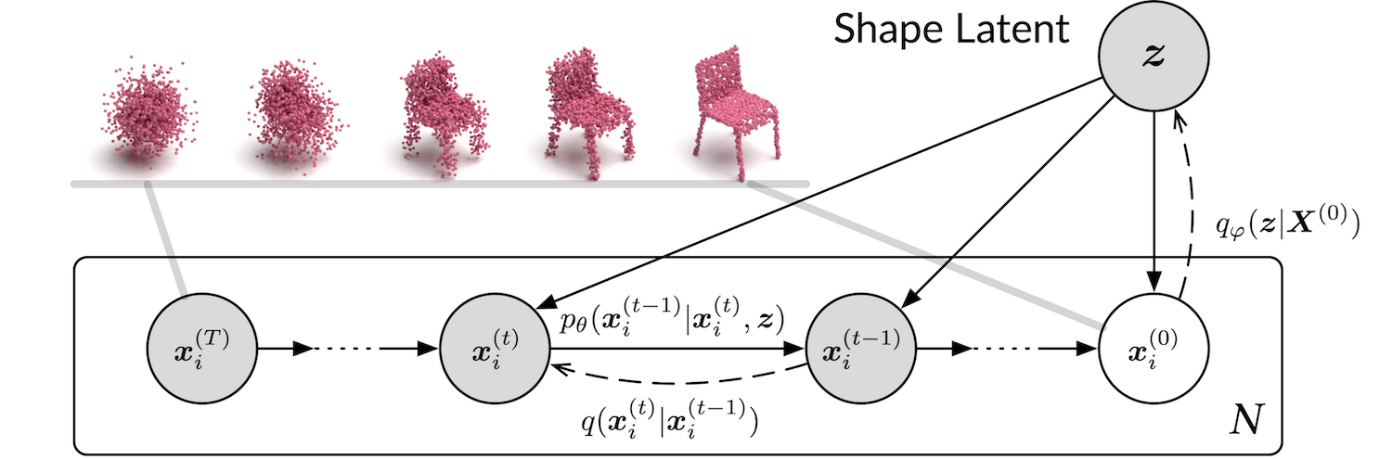


Figure 2.4.3.1: Parti System Architecture

CMRCET B. Tech (CSE) Page No 13

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 3**

**PROPOSED**

**SYSTEM**

CMRCET B. Tech (CSE) Page No 14

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 3**

**PROPOSED SYSTEM**

**3.1.** **Objective of Proposed Model**:

The objective of the "Generative Neural Networks Novel Image Generation" project is threefold:

**Expand Creative and Generative Capabilities**:

The project aims to push the boundaries of neural networks beyond traditional discriminative models by focusing on generative models. It seeks to enable neural networks to create entirely new and unique creative content, thus expanding their creative and generative capabilities.

**Address Data Scarcity Through Data Augmentation**:

By leveraging generative models, the project seeks to address data scarcity issues in various domains by generating synthetic content. This synthetic content can augment real datasets, enhancing machine learning model performance in scenarios where access to authentic data is limited.

**Facilitate Innovation Across Industries**:

The project aims to have a broad impact across multiple industries such as healthcare, fashion, and data security. It assists in medical image analysis by generating realistic data for algorithm training, aids in fashion design by creating new patterns and styles, and addresses data privacy concerns by enabling information sharing without compromising sensitive details. Overall, the project endeavors to bridge technology and creativity, fostering innovation in data science and beyond.

CMRCET B. Tech (CSE) Page No 15

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**3.2.** **Algorithms Used for Proposed Model:**

In the "Generative Neural Networks Novel Image Generation" project, several algorithms are employed to train and optimize the generative neural network architecture, primarily focusing on Generative Adversarial Networks (GANs). Here's an overview of the algorithms used:

## GANs

[GANs](https://arxiv.org/abs/1406.2661)are a powerful recent innovation in the domain of deep learning and they are used for unsupervised learning. GANs consist of two models, namely, the generative model and the discriminator model. The generative model is responsible for creating fake data instances that resemble your training data. On the other hand, the discriminator model behaves as a classifier that distinguishes between real data instances from the output of the generator.

The generator attempts to deceive the discriminator by generating real images as far as possible, and the discriminator tries to keep from being deceived. The discriminator penalizes the generator for producing an absurd output. At the initial stages of the training process, the generator generates fake data, and the discriminator quickly learns to tell that it’s fake.

But as the training progresses, the generator moves closer to producing an output that can fool the discriminator. Finally, if generator training goes well, then the discriminator performance gets worse because it can’t quickly tell the difference between real and fake. It starts to classify the fake data as real, and its accuracy decreases. Below is a picture of the whole system:

CMRCET B. Tech (CSE) Page No 16

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

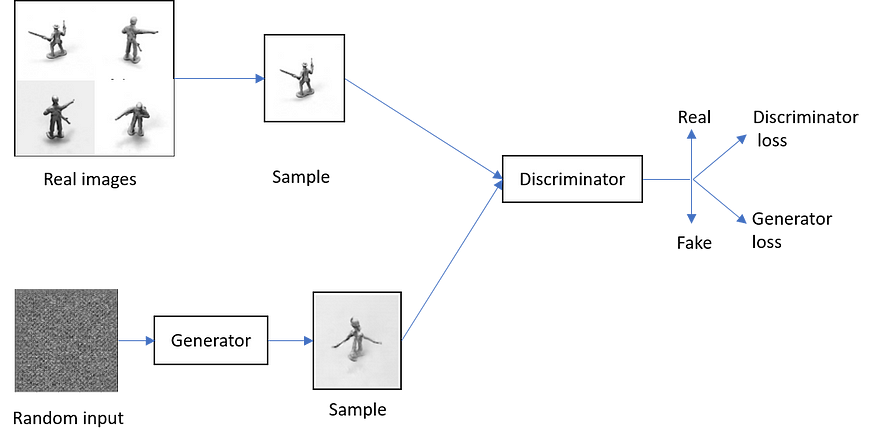


Figure 3.2.1: GAN framework

## DCGAN

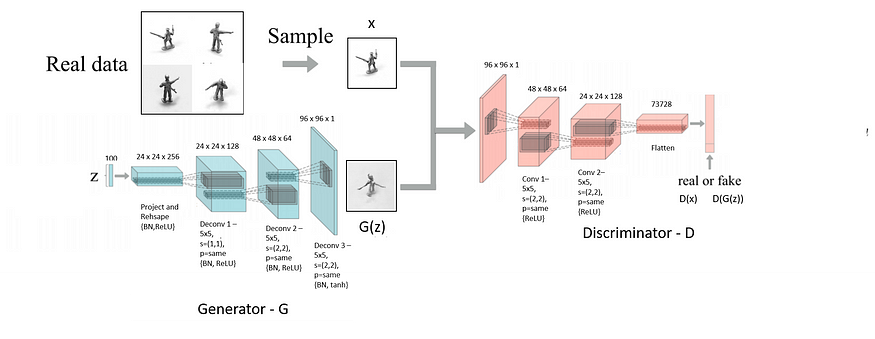
[DCGANs](https://arxiv.org/abs/1511.06434) are very similar to GANs but specifically focuses on using deep convolutional neural networks (CNN, or ConvNet) in place of fully-connected networks which are used in vanilla GANs. [CNNs](https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf) are commonly used in the domain of pattern recognition within images. 

Figure 3.2.2: DCGAN architecture

CMRCET B. Tech (CSE) Page No 17

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

## CAPSGAN

Deep learning techniques like CNN have outperformed classical machine learning techniques for the image classification task. However, a new network called [CapsNet](https://www.cs.toronto.edu/~hinton/absps/transauto6.pdf) has outperformed CNNs for this task. The limitation of a CNN for the object recognition task is that the activation of its neurons is based on the chances of detecting specific image features.

Neurons do not consider the properties or features of an image, such as pose, texture, and deformation of the objects in the image. In other words, CNNs are incapable because of their invariance as a result of the pooling operation. Basically, a capsule is a group of neural layers. While a typical neuron outputs a single scalar value, a capsule outputs a vector representing a generalized set of related object properties.

A capsule attempts to capture many object properties like pose (position, angle of view, size), deformation, and the texture inside an image to define the probability of some object existence. Transforming autoencoders (refer to AEGAN section for a brief introduction to autoencoders) make use of complex feature detectors called a capsule, to capture the exact pose of each feature in the image and this is how they try to learn the overall transformation matrix.

Capsules allow the autoencoder to maintain translational invariance without throwing away important positional information. They are not only capable of recognizing features in different poses and lighting conditions but are also capable of outputting pose-specific variables to be used by higher visual layers rather than discarding them.

The goal was not to recognize objects in images but to accept an image and its pose as input and output the same image in the original pose.

CMRCET B. Tech (CSE) Page No 18

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

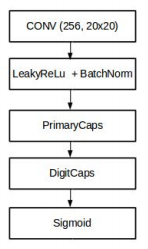


Figure 3.2.3: [CapsGAN discriminator architecture](https://www.vdu.lt/cris/bitstream/20.500.12259/36302/3/huseyn_gardirov_md.pdf)

## AEGAN

Autoencoding Generative Adversarial Networks (AEGAN) is a four-network model comprising of two GANs and two autoencoders as shown below:

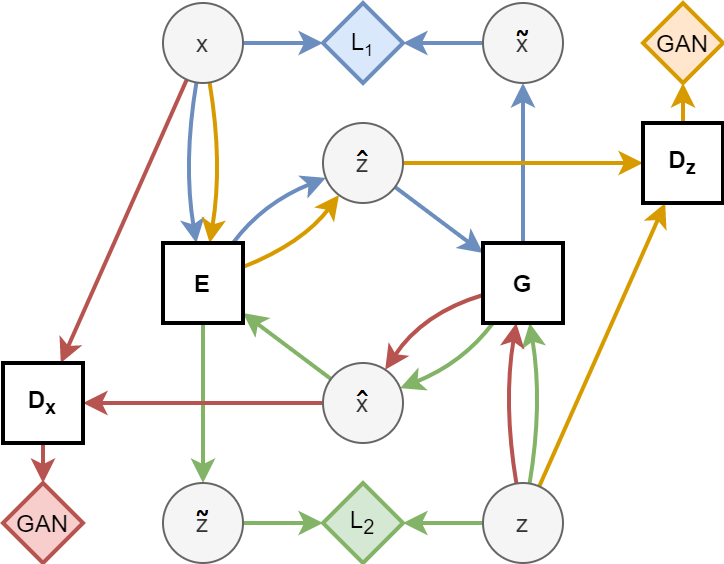


Figure 3.2.4: AEGAN Architecture

CMRCET B. Tech (CSE) Page No 19

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

Just like GANs, autoencoders are a type of unsupervised learning algorithms. The autoencoders consist of two virtual components in its network, namely, the encoder model and the decoder model. The encoder model maps the input data to the network’s internal representation, just like the notion of data compression operation, and the decoder model tries to reconstruct the input from the network’s internal data representation just like the notion of data decompression operation. Therefore, the output shape of the autoencoder is the same as the input, that allows the network to learn basic representations better.

AEGAN leverages the advantages of GANs and autoencoders by stabilizing the GAN training and thereby overcomes the common problems of GANs, namely, mode collapse and lack of convergence.

Here is a brief discussion about the algorithms used in our model training:

**Generative Adversarial Networks (GANs)**

Generative Adversarial Networks (GANs) was first introduced by Ian Goodfellow in 2014. GANs are a powerful class of neural networks that are used for unsupervised learning. GANs can create anything whatever you feed to them, as it Learn-Generate-Improve. To understand GANs first you must have little understanding of Convolutional Neural Networks. CNNs are trained to classify images with respect to their labels if an image is fed to a CNN, it analyzes the image pixel by pixel and is passed through nodes present in CNN’s hidden layers and as an output, it tells what the image is about or what it sees in the image. For example: If a CNN is trained to classify dogs and cats and an image is fed to this CNN, it can tell whether there is a dog or a cat in that image. Therefore it can also be called as a classification algorithm. How GANs are different? GANs can be divided into two parts which are the Generator and the Discriminator. Discriminator – This part of GANs can be considered similar to what CNNs does. Discriminator is a Convolutional Neural Network consisting of many hidden layers and one output layer, the major difference here is the output layer of GANs can have only two outputs, unlike CNNs, which can have outputs respect to the number of labels it is trained on.

CMRCET B. Tech (CSE) Page No 20

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

The output of the discriminator can either be 1 or 0 because of a specifically chosen activation function for this task, if the output is 1 then the provided data is real and if the output is 0 then it refers to it as fake data. Discriminator is trained on the real data so it learns to recognize how actual data looks like and what features should the data have to be classified as real.

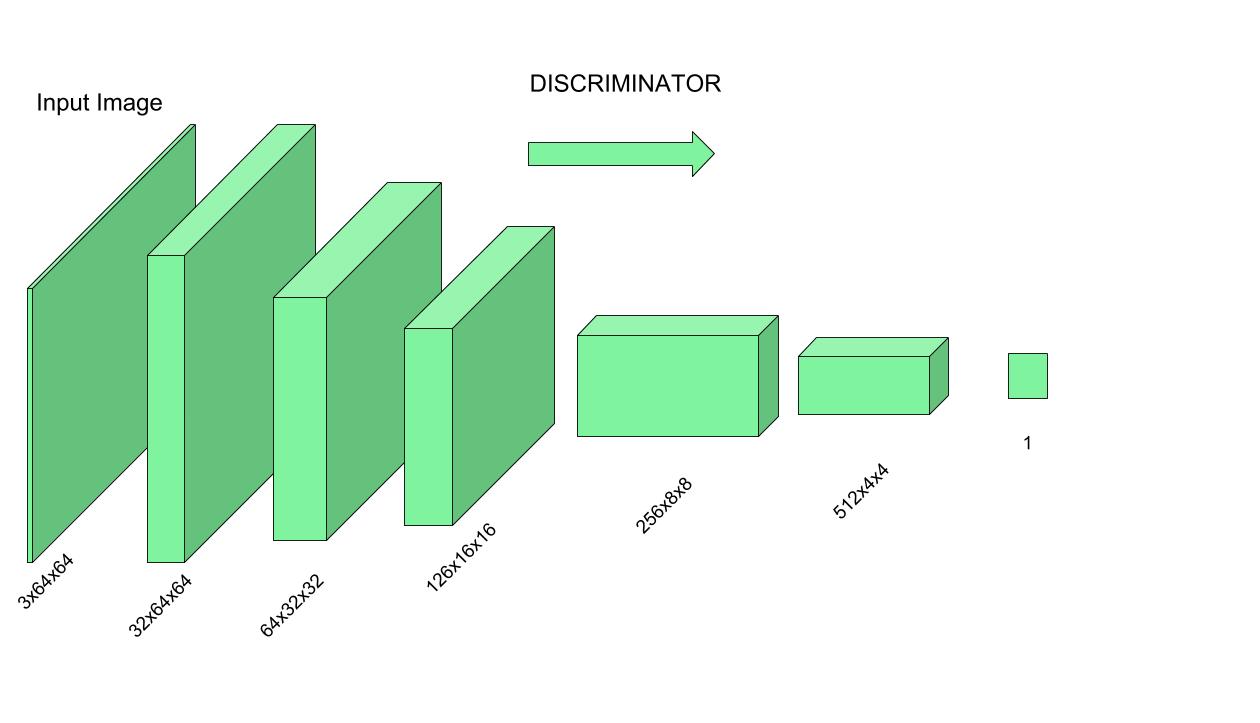


Figure 3.2.5: GAN Discriminator

Generator – From the name itself, we can understand that it’s a generative algorithm. Generator is an Inverse Convolutional Neural Net, it does exactly opposite of what a CNN does, because in CNN an actual image is given as an input and a classified label is expected as an output but in Generator, a random noise (a vector having some values to be precise) is given as an input to this Inverse CNN and an actual image is expected as an output. In simple terms, it generates data from a piece of data using its own imagination.

CMRCET B. Tech (CSE) Page No 21

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

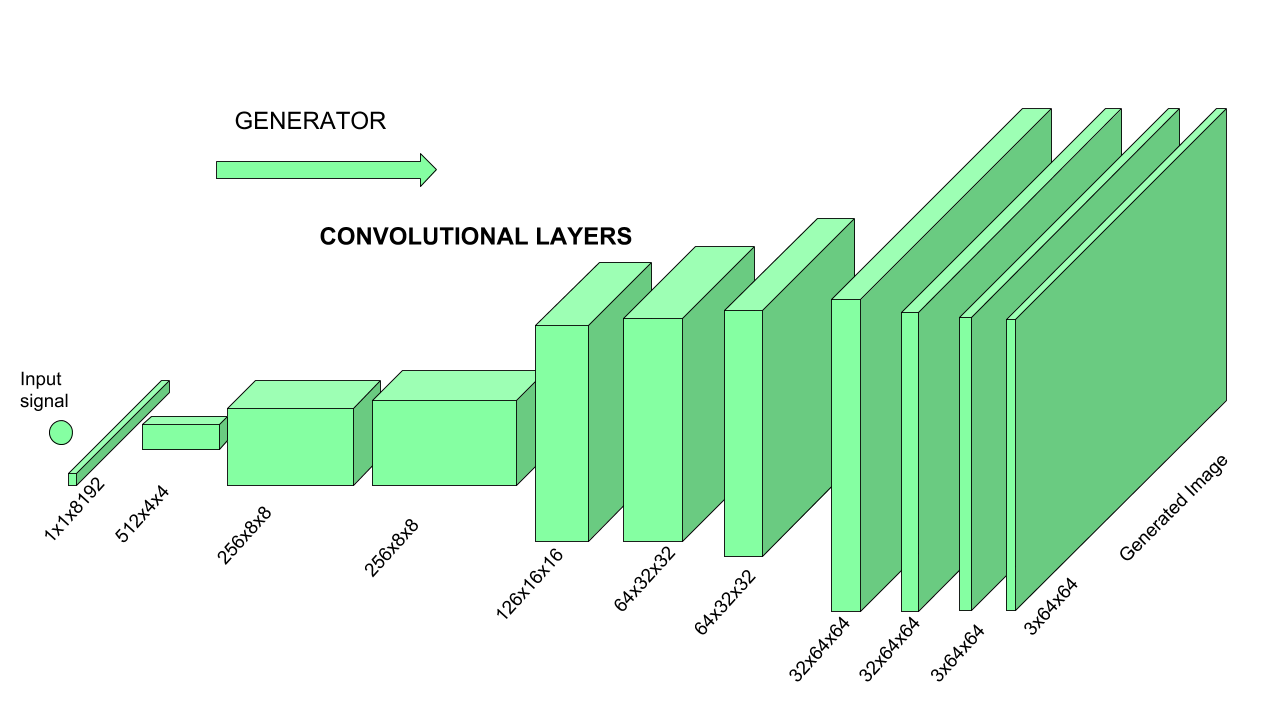


Figure 3.2.6: GAN Generator

As shown in the above image, a random value vector is given as input to Inverse-CNN and after getting passed through the hidden layers and activation functions an image is received as the output. Working of both Generator and Discriminator together: As we already discussed Discriminator is trained on actual data to classify whether given data is true or not, so Discriminator’s work is to tell what’s real and what’s fake. Now the Generator starts to generate data from a random input and then that generated data is passed to Discriminator as input now Discriminator analyzes the data and checks how close it is to be classified as real, if the generated data does not contain enough features to be classified as real by the Discriminator, then this data and weights associated with it are sent back to the Generator using backpropagation, so that it can readjust the weights associated with the data and create new data which is better than the previous one. This freshly generated data is again passed to the Discriminator and it continues. This process keeps repeating as long as the Discriminator keeps classifying the generated data as fakes, for every time data is classified as fake and with every

CMRCET B. Tech (CSE) Page No 22

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

backpropagation the quality of data keeps getting better and better and there comes a time when the Generator becomes so accurate that it becomes tough to distinguish between the real data and the data generated by the Generator.

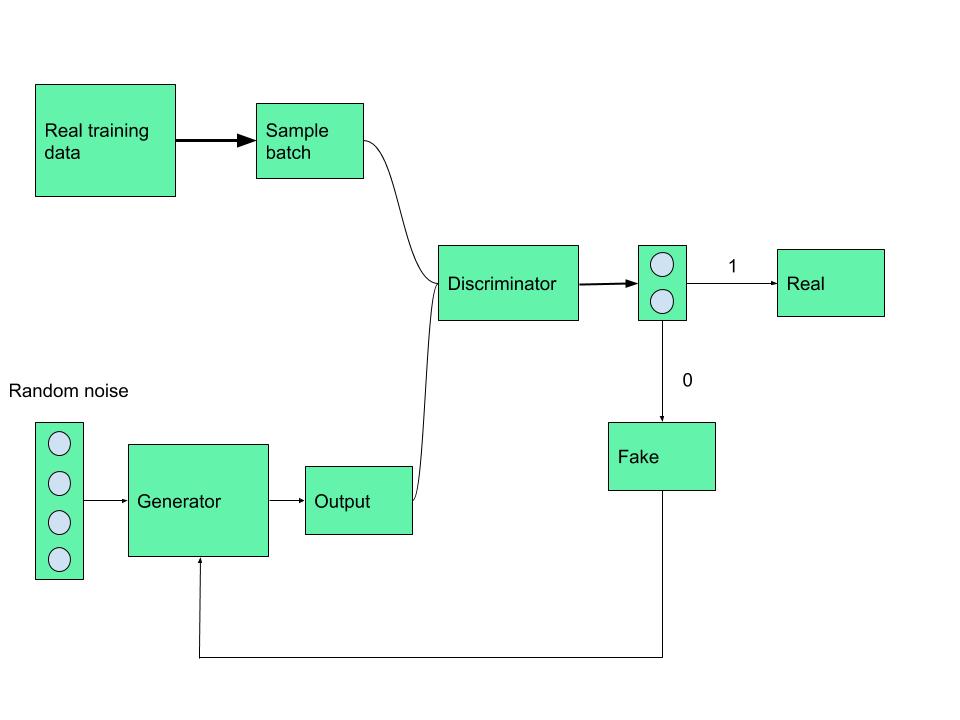


Figure 3.2.7: GAN Discriminator and Generator

A key element responsible for creating fresh, accurate data in a Generative Adversarial Network (GAN) is the generator model. The generator takes random noise as input and converts it into complex data samples, such text or images. It is commonly depicted as a deep neural network.

CMRCET B. Tech (CSE) Page No 23

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

The training data’s underlying distribution is captured by layers of learnable parameters in its design through training. The generator adjusts its output to produce samples that closely mimic real data as it is being trained by using backpropagation to fine-tune its parameters.

The generator’s ability to generate high-quality, varied samples that can fool the discriminator is what makes it successful.

**Generator Loss**

The objective of the generator in a GAN is to produce synthetic samples that are realistic enough to fool the discriminator. The generator achieves this by minimizing its loss function ​. The loss is minimized when the log probability is maximized, i.e., when the discriminator is highly likely to classify the generated samples as real.

An artificial neural network called a discriminator model is used in Generative Adversarial Networks (GANs) to differentiate between generated and actual input. By evaluating input samples and allocating probability of authenticity, the discriminator functions as a binary classifier.

Over time, the discriminator learns to differentiate between genuine data from the dataset and artificial samples created by the generator. This allows it to progressively hone its parameters and increase its level of proficiency.

Convolutional layers or pertinent structures for other modalities are usually used in its architecture when dealing with picture data. Maximizing the discriminator’s capacity to accurately identify generated samples as fraudulent and real samples as authentic is the aim of the adversarial training procedure. The discriminator grows increasingly discriminating as a result of the generator and discriminator’s interaction, which helps the GAN produce extremely realistic-looking synthetic data overall.

CMRCET B. Tech (CSE) Page No 24

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**Discriminator Loss**

The discriminator reduces the negative log likelihood of correctly classifying both produced and real samples. This loss incentivizes the discriminator to accurately categorize generated samples as fake and real samples with the following equation

**The steps involved in how a GAN works:**

Initialization: Two neural networks are created: a Generator (G) and a Discriminator (D).

Generator is tasked with creating new data, like images or text, that closely resemble real data.

Discriminator acts as a critic, trying to distinguish between real data (from a training dataset) and the data generated by the Generator.Generator’s First Move: Generator takes a random noise vector as input. This noise vector contains random values and acts as the starting point for Generator’s creation process. Using its internal layers and learned patterns, G transforms the noise vector into a new data sample, like a generated image.

Discriminator’s Turn: Discriminator receives two kinds of inputs:

Real data samples from the training dataset. The data samples generated by Generator in the previous step. Discriminator’s job is to analyze each input and determine whether it’s real data or something Generator cooked up. It outputs a probability score between 0 and 1. A score of 1 indicates the data is likely real, and 0 suggests it’s fake. The Learning Process: Now, the adversarial part comes in:

If Discriminator correctly identifies real data as real (score close to 1) and generated data as fake (score close to 0), both Generator and Discriminator are rewarded to a small degree. This is because they’re both doing their jobs well. However, the key is to continuously improve. If Discriminator consistently identifies everything correctly, it won’t learn much. So, the goal is for G to eventually trick Discriminator.

When Discriminator mistakenly labels Generator’s creation as real (score close to 1), it’s a sign that Generator is on the right track.

CMRCET B. Tech (CSE) Page No 25

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**Generator’s Improvement:**

In this case, Generator receives a significant positive update, while Discriminator receives a penalty for being fooled. Conversely, if Discriminator correctly identifies Generator’s fake data (score close to 0), but Generator receives no reward, Discriminator is further strengthened in its discrimination abilities.

This ongoing duel between Generator and Discriminator refines both networks over time.

As training progresses, Generator gets better at generating realistic data, making it harder for Discriminator to tell the difference. Ideally, Generator becomes so adept that Discriminator can’t reliably distinguish real from fake data. At this point, Generator is considered well-trained and can be used to generate new, realistic data samples.

GANs consists of two neural networks. There is a Generator G(x) and a Discriminator D(x). Both of them play an adversarial game. The generator's aim is to fool the discriminator by producing data that are similar to those in the training set. The discriminator will try not to be fooled by identifying fake data from real data. Both of them work simultaneously to learn and train complex data like audio, video, or image files. The Generator network takes a sample and generates a fake sample of data. The Generator is trained to increase the Discriminator network's probability of making mistakes.

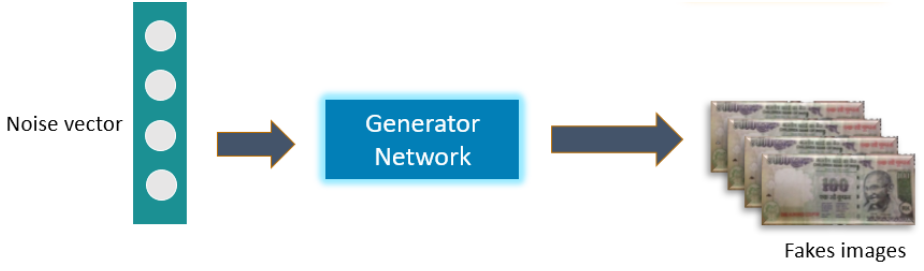


Figure 3.2.8: Generated fake images

CMRCET B. Tech (CSE) Page No 26

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

Below is an example of a GAN trying to identify if the 100 rupee notes are real or fake. So, first, a noise vector or the input vector is fed to the Generator network. The generator creates fake 100 rupee notes. The real images of 100 rupee notes stored in a database are passed to the discriminator along with the fake notes. The Discriminator then identifies the notes as classifying them as real or fake.

We train the model, calculate the loss function at the end of the discriminator network, and backpropagate the loss into both discriminator and generator models.

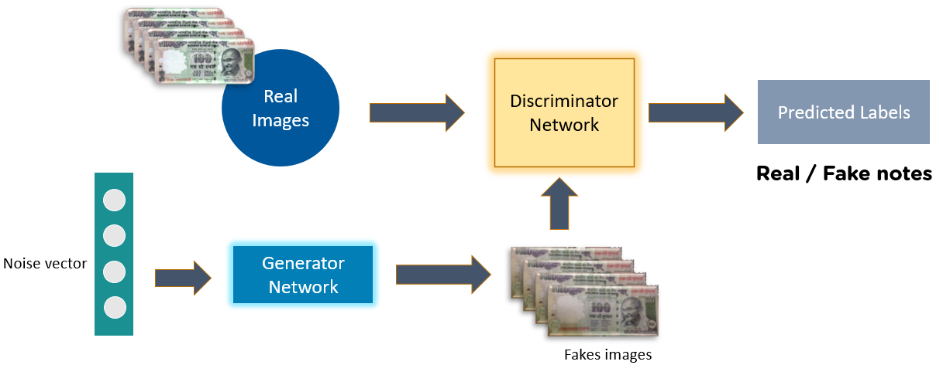


Figure 3.2.9: GAN Framework

### Mathematical Equation

The mathematical equation for training a GAN can be represented as:

Here,

G = Generator

CMRCET B. Tech (CSE) Page No 27

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

D = Discriminator

Pdata(x) = distribution of real data

p(z) = distribution of generator

x = sample from Pdata(x)

z = sample from P(z) D(x) Discriminator network G(z) = Generator network

With this understanding, let’s learn the next topic on what are GANs, i.e. training a GAN.

**Steps for Training GAN**

* Define the problem
* Choose the architecture of GAN
* Train discriminator on real data
* Generate fake inputs for the generator
* Train discriminator on fake data
* Train generator with the output of the discriminator

Let us now look at the different types of GANs.

Vanilla GANs: Vanilla GANs have a min-max optimization formulation where the Discriminator is a binary classifier and uses sigmoid cross-entropy loss during optimization. The Generator and the Discriminator in Vanilla GANs are multi-layer perceptrons. The algorithm tries to optimize the mathematical equation using stochastic gradient descent.

CMRCET B. Tech (CSE) Page No 28

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

Deep Convolutional GANs (DCGANs): DCGANs support convolution neural networks instead of vanilla neural networks at both Discriminator and Generator. They are more stable and generate better quality images. The Generator is a set of convolution layers with fractional-strided convolutions or transpose convolutions, so it up-samples the input image at every convolutional layer.

Conditional GANs: Vanilla GANs can be extended into Conditional models by using extra-label information to generate better results. In CGAN, an additional parameter ‘y’ is added to the Generator for generating the corresponding data. Labels are fed as input to the Discriminator to help distinguish the real data from the fake generated data.

**3.3.** **Designing:**

The design of the "Generative Neural Networks Novel Image Generation" project involves several key components to ensure its success in achieving its objectives. Here's an outline of the project's design:

1. **Project Scope and Objectives:**

- Define the scope of the project, including its objectives and intended outcomes.

- Establish clear goals for expanding the creative and generative capabilities of neural networks, addressing data scarcity through data augmentation, and facilitating innovation across industries.

2. **Selection of Neural Network Architecture:**

- Choose an appropriate neural network architecture, primarily focusing on Generative Adversarial Networks (GANs) for image generation.

CMRCET B. Tech (CSE) Page No 29

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

3. **Data Acquisition and Preprocessing:**

- Gather diverse datasets suitable for training GANs, ensuring they encompass a wide range of styles and characteristics.

- Implement preprocessing techniques to standardize the data, including resizing, normalization, and augmentation.

- Ensure the consistency and quality of the dataset to optimize the training process.

4. **Model Training and Optimization:**

- Implement the training pipeline for the GAN architecture, consisting of the generator and discriminator networks.

- Utilize adversarial training techniques to iteratively improve the quality of generated images.

- Monitor the training process and adjust hyperparameters as necessary to optimize performance.

5. **Evaluation and Validation:**

- Assess the quality, diversity, and realism of the generated images using evaluation metrics such as Fréchet Inception Distance (FID) or domain-specific measures.

- Validate the effectiveness of the trained GAN architecture through visual inspection and comparison with real images.

- Iterate on the model based on evaluation results to refine its performance.

6. **Fine-tuning and Iterative Refinement:**

- Apply fine-tuning techniques to further enhance the performance of the GAN architecture.

- Explore novel approaches and architectural modifications to improve the creativity and diversity of the generated images.

CMRCET B. Tech (CSE) Page No 30

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**3.3.1.UML Diagram**

Novel Image Generation

Evaluate Generated Images

Train Generative Models

Assess Quality, Diversity, and Realism

Preprocess Data  
(Resize, Normalize, Augment)

Fine-tune Model (Iterative Refinement)

Visual Inspection and comparison with Real Images

- The "Novel Image Generation" component represents the overarching project.

- The first step involves training the generative models, which branches into preprocessing the data and then training the models themselves.

CMRCET B. Tech (CSE) Page No 31

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

- Once the models are trained, they undergo evaluation, branching into assessing the quality, diversity, and realism of the generated images.

- Finally, there are two branches for further refinement: fine-tuning the model through iterative refinement and conducting visual inspection and comparison with real images.

**3.4.** **Stepwise Implementation and Code:**

**1. Imports:**

- Import necessary libraries such as TensorFlow, TensorFlow Datasets, NumPy, Seaborn, PIL (Python Imaging Library), time, imageio, glob, IPython, and datetime.

- These libraries are used for tasks like machine learning (TensorFlow), dataset handling (TensorFlow Datasets), numerical computations (NumPy), visualization (Seaborn and Matplotlib), working with images (PIL), time operations, image input-output, file handling, and printing timestamps.

**2. Device Check:**

- Check for available devices using TensorFlow's device\_lib.

- Verify if a GPU is available for TensorFlow operations.

**3. Timer Function:**

- Define a timer function to calculate the time elapsed since its invocation.

- Use it to time the execution of code blocks.

**4. Changing Directory:**

- Change the current working directory to '/content'. This might be specific to Google Colab.

CMRCET B. Tech (CSE) Page No 32

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**5. Loading Dataset:**

- Load the 'smallnorb' dataset from TensorFlow Datasets.

- Split the dataset into training and testing sets.

- Shuffle the files.

- Load the data as supervised (input-output pairs).

- Print information about the loaded dataset.

**6. Plotting Dataset:**

- Define a function named 'plot\_dataset' to visualize the dataset.

- Iterate over a subset of examples from the dataset, plotting images with corresponding labels.

- Each subplot shows an image with its label.

**7. Normalization Function:**

- Define a function named 'normalize\_img' to normalize images and labels.

- Cast the image to float32.

- Resize the image to the specified width and height.

- Normalize the image pixel values to the range [-1, 1].

**8. Constants:**

- Set constant values for the width and height of downsampled images, buffer size, and batch size.

**9. Dataset Preprocessing:**

- Normalize the training dataset and cache it to memory.

- Print the length of the dataset.

- Shuffle and batch the dataset.

CMRCET B. Tech (CSE) Page No 33

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**10. Testing Dataset Preprocessing:**

- Normalize the testing dataset.

- Filter the dataset to exclude certain label categories.

- Batch, cache, and prefetch the dataset.

**11. Generator Model:**

- Define the generator model using TensorFlow's Sequential API.

- Use dense and convolutional layers to generate images from random noise.

- Take a 100-dimensional noise vector as input and output a 96x96 grayscale image.

**12. Generating Sample Image:**

- Generate a sample image using the generator model.

- Sample random noise from a normal distribution and pass it through the generator.

**13. Discriminator Model:**

- Define the discriminator model using TensorFlow's Sequential API.

- Use convolutional layers followed by fully connected layers for binary classification (real or fake).

**14. Testing Discriminator:**

- Pass the generated image through the discriminator to check its output.

- Ensure that the discriminator is functioning correctly.

**15. Loss Functions:**

- Define the loss functions for the generator and discriminator.

- Use binary cross-entropy loss for both.

CMRCET B. Tech (CSE) Page No 34

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**16. Optimizer:**

- Set up Adam optimizers for both the generator and discriminator.

**17. Checkpointing:**

- Set up a checkpoint directory to save model checkpoints during training.

**18. Training Loop:**

- Define a `train\_step` function to perform one step of training.

- Iterate over the dataset batches within each epoch.

- Compute gradients and update model parameters.

- Record generator and discriminator losses for each epoch.

- Clear the output display and generate sample images for visualization at the end of each epoch.

**19. Save Checkpoints:**

- Save model checkpoints periodically during training.

**20. Generate and Save Images:**

- Define a function to generate and save sample images for visualization.

**21. Training on Specific Class:**

- Provide instructions for training the model on a particular class.

- Filter the dataset accordingly and adjust buffer size and batch size.

**22. Training:**

- Initiate the training process using the defined training loop function.

- Record generator and discriminator losses for each epoch.

CMRCET B. Tech (CSE) Page No 35

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**23. Plotting Losses:**

- Plot the generator and discriminator losses across epochs.

**24. Creating GIF Animation:**

- Install the `imageio` library if not already installed.

- Create an animated GIF from the saved images using imageio.

**25. Downloading GIF and PNGs:**

- Download the generated GIF animation and PNG images.

**26. Zipping Training Checkpoints:**

- Zip the training checkpoints directory.

**27. Downloading Training Checkpoints Zip:**

- Download the zipped training checkpoints.

**28. Importing Libraries for GAN Implementation:**

- Import necessary libraries for implementing a Generative Adversarial Network (GAN) using TensorFlow.

**29. Defining Discriminator and Generator Models:**

- Define functions to create and compile the discriminator and generator models.

**30. Training GAN Model:**

- Define functions for loading real samples, generating fake samples, and training the GAN model.

- Iterate over training steps, update model parameters, and print losses.

CMRCET B. Tech (CSE) Page No 36

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CODE:**

import tensorflow as tf # '2.2.0'

import tensorflow\_datasets as tfds

from tensorflow import keras

from tensorflow.keras import layers

# Helper libraries

import numpy as np

import seaborn as sns

import os

import PIL

import time

import imageio

import glob

from IPython import display

from datetime import datetime

# visualization tools

%matplotlib inline

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings("ignore")

# device check

from tensorflow.python.client import device\_lib

print('Devices:', device\_lib.list\_local\_devices())

# GPU check

if not tf.test.gpu\_device\_name():

CMRCET B. Tech (CSE) Page No 37

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

print('No GPU found.')

else:

print('Default GPU Device: {}'.format(tf.test.gpu\_device\_name()))

def timer(start\_time=None):

if not start\_time:

start\_time = datetime.now()

return start\_time

elif start\_time:

thour, temp\_sec = divmod((datetime.now() - start\_time).total\_seconds(), 3600)

tmin, tsec = divmod(temp\_sec, 60)

print('Time taken: %i hours %i minutes and %s seconds.' % (thour, tmin, round(tsec, 2)))

os.chdir('/content') # Google Colab - location to save or query files

# https://www.tensorflow.org/datasets/api\_docs/python/tfds/load#args

# https://www.tensorflow.org/datasets/catalog/smallnorb

(ds\_train, ds\_test), ds\_info = tfds.load(

'smallnorb',

split=['train', 'test'],

shuffle\_files=True,

as\_supervised=True,

with\_info=True,

)

# Print the loaded dataset information

print(ds\_info)

DOWNSAMPLE\_IMAGE\_WIDTH = 96

DOWNSAMPLE\_IMAGE\_HEIGHT = 96

def plot\_dataset(ds\_info, ds, rows=5, columns=5):

fig=plt.figure(figsize=(21, 21))

CMRCET B. Tech (CSE) Page No 38

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

i = 1

switcher = {}

# <https://github.com/tensorflow/datasets/blob/master/README.md>

# https://www.tensorflow.org/datasets/overview

for idx in range(ds\_info.features['label\_category'].num\_classes):

switcher[idx] = ds\_info.features['label\_category'].int2str(idx)

# ax enables access to manipulate each of subplots

ax = []

for example in ds.take(columns\*rows): # Only take a single example

image, label = example[0], example[1]

# create subplot and append to ax

ax.append( fig.add\_subplot(rows, columns, i) )

ax[-1].set\_title("label\_category: {0} - {1}".format(label.numpy(), switcher.get(label.numpy()))) # set title

plt.imshow(image.numpy()[:, :, 0].astype(np.float32), cmap=plt.get\_cmap("gray"))

plt.axis('off')

i += 1

plt.show()

plot\_dataset(ds\_info, ds\_train)

plot\_dataset(ds\_info, ds\_test)

BUFFER\_SIZE = 24300 # Individual class size: 4860 # Total dataset size : 24300

BATCH\_SIZE = 8

def normalize\_img(image, label\_category):

image = tf.cast(image, tf.float32) # TFDS provide the images as tf.uint8, while the model expect tf.float32, so normalize images

image = tf.image.resize(image, (DOWNSAMPLE\_IMAGE\_WIDTH, DOWNSAMPLE\_IMAGE\_HEIGHT))

image = (image - 127.5) / 127.5 # Normalize the images to [-1, 1]

CMRCET B. Tech (CSE) Page No 39

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

return image, label\_category

ds\_train\_nrm = ds\_train.map(

normalize\_img, num\_parallel\_calls=tf.data.experimental.AUTOTUNE)

# Take only label\_category 4 (cars) for image generation if required, also change BUFFER\_SIZE accordingly

#ds\_train\_nrm = ds\_train\_nrm.filter(lambda x,y: tf.reduce\_all(tf.not\_equal(y, [0,1,2,3])))

# Select few images if needed

#ds\_train\_nrm = ds\_train\_nrm.take(100)

ds\_train\_nrm = ds\_train\_nrm.cache()

print("Length of dataset: ", ds\_train\_nrm.reduce(0, lambda x, \_: x + 1).numpy())

ds\_train\_nrm = ds\_train\_nrm.shuffle(ds\_train\_nrm.reduce(0, lambda x, \_: x + 1).numpy()) # shuffle(length\_of\_dataset)

ds\_train\_nrm = ds\_train\_nrm.batch(BATCH\_SIZE)

ds\_train\_nrm = ds\_train\_nrm.prefetch(tf.data.experimental.AUTOTUNE)

ds\_test\_nrm = ds\_test.map(

normalize\_img, num\_parallel\_calls=tf.data.experimental.AUTOTUNE)

ds\_test\_nrm = ds\_test\_nrm.filter(lambda x,y: tf.reduce\_all(tf.not\_equal(y, [0,1,2,3])))

ds\_test\_nrm = ds\_test\_nrm.batch(BATCH\_SIZE)

ds\_test\_nrm = ds\_test\_nrm.cache()

ds\_test\_nrm = ds\_test\_nrm.prefetch(tf.data.experimental.AUTOTUNE)

def make\_generator\_model():

model = tf.keras.Sequential()

# project

model.add(layers.Dense(24\*24\*256, use\_bias=False, input\_shape=(100,)))

model.add(layers.BatchNormalization())

model.add(layers.LeakyReLU(alpha=0.2))

# reshape

model.add(layers.Reshape((24, 24, 256)))

assert model.output\_shape == (None, 24, 24, 256) # Note: None is the batch size

CMRCET B. Tech (CSE) Page No 40

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

# upsample or deconv1

model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use\_bias=False))

assert model.output\_shape == (None, 24, 24, 128)

model.add(layers.BatchNormalization())

model.add(layers.LeakyReLU(alpha=0.2))

# upsample or deconv2

model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use\_bias=False))

assert model.output\_shape == (None, 48, 48, 64)

# scale from [0,255] to [-1,1]

X = (X - 127.5) / 127.5

print(X.shape, y\_train.shape)

return [X, y\_train]

# select a supervised subset of the dataset

def select\_supervised\_samples(dataset, n\_samples=100, n\_classes=5):

X, y = dataset

X\_list, y\_list = list(), list()

class\_list = [4] # for filtering, for entire dataset = [0,1,2,3,4]

n\_classes = len(class\_list)

n\_per\_class = int(n\_samples / n\_classes)

for i in class\_list:

# get all images for this class

X\_with\_class = X[y == i]

# choose rando3m instances

ix = randint(0, len(X\_with\_class), n\_per\_class)

# add to list

[X\_list.append(X\_with\_class[j]) for j in ix]

[y\_list.append(i) for j in ix]

CMRCET B. Tech (CSE) Page No 41

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

return asarray(X\_list), asarray(y\_list)

# select real samples

def generate\_real\_samples(dataset, n\_samples):

# split into images and labels

images, labels = dataset

# choose random instances

ix = randint(0, images.shape[0], n\_samples)

# select images and labels

X, labels = images[ix], labels[ix]

# generate class labels

y = ones((n\_samples, 1))

return [X, labels], y

# generate points in latent space as input for the generator

def generate\_latent\_points(latent\_dim, n\_samples):

# generate points in the latent space

z\_input = randn(latent\_dim \* n\_samples)

# reshape into a batch of inputs for the network

z\_input = z\_input.reshape(n\_samples, latent\_dim)

return z\_input

# use the generator to generate n fake examples, with class labels

def generate\_fake\_samples(generator, latent\_dim, n\_samples):

# generate points in latent space

z\_input = generate\_latent\_points(latent\_dim, n\_samples)

# predict outputs

images = generator.predict(z\_input)

# create class labels

y = zeros((n\_samples, 1))

return images, y

# generate samples and save as a plot and save the model

CMRCET B. Tech (CSE) Page No 42

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

def summarize\_performance(step, g\_model, latent\_dim, dataset, n\_samples=25):

# prepare fake examples

X, \_ = generate\_fake\_samples(g\_model, latent\_dim, n\_samples)

fig = pyplot.figure(figsize=(21,21))

# plot images

for i in range(n\_samples):

# define subplot

pyplot.subplot(5, 5, 1 + i)

# turn off axis

pyplot.axis('off')

# plot raw pixel data

pyplot.imshow(X[i, :, :, 0] \* 127.5 + 127.5, cmap='gray')

# save plot to file

filename1 = 'generated\_plot\_%04d.png' % (step+1)

pyplot.savefig(filename1)

pyplot.close()

# evaluate the classifier model

X, y = dataset

# save the generator model

filename2 = 'g\_model\_%04d.h5' % (step+1)

g\_model.save(filename2)

print('>Saved: %s, and %s' % (filename1, filename2))

# train the generator and discriminator

def train(g\_model, d\_model, gan\_model, dataset, latent\_dim, n\_epochs=200, n\_batch=64):

# select supervised dataset

X\_sup, y\_sup = select\_supervised\_samples(dataset)

print(X\_sup.shape, y\_sup.shape)

# calculate the number of batches per training epoch

CMRCET B. Tech (CSE) Page No 43

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

bat\_per\_epo = int(dataset[0].shape[0] / n\_batch)

# calculate the number of training iterations

n\_steps = bat\_per\_epo \* n\_epochs

# calculate the size of half a batch of samples

half\_batch = int(n\_batch / 2)

print('n\_epochs=%d, n\_batch=%d, half\_batch=%d, batch\_per\_epoch=%d, n\_steps=%d' % (n\_epochs, n\_batch, half\_batch, bat\_per\_epo, n\_steps))

# manually enumerate epochs

for i in range(n\_steps):

# update unsupervised discriminator (d)

[X\_real, \_], y\_real = generate\_real\_samples(dataset, half\_batch)

d\_loss1 = d\_model.train\_on\_batch(X\_real, y\_real)

X\_fake, y\_fake = generate\_fake\_samples(g\_model, latent\_dim, half\_batch)

d\_loss2 = d\_model.train\_on\_batch(X\_fake, y\_fake)

# update generator (g)

X\_gan, y\_gan = generate\_latent\_points(latent\_dim, n\_batch), ones((n\_batch, 1))

g\_loss = gan\_model.train\_on\_batch(X\_gan, y\_gan)

# summarize loss on this batch

print('>%d, d[%.3f,%.3f], g[%.3f]' % (i+1, d\_loss1, d\_loss2, g\_loss))

# evaluate the model performance every so often

if (i+1) % (bat\_per\_epo \* 1) == 0:

summarize\_performance(i, g\_model, latent\_dim, dataset)

latent\_dim = 100

d\_model = define\_discriminator()

g\_model = define\_generator(latent\_dim)

gan\_model = define\_gan(g\_model, d\_model)

dataset = load\_real\_samples()

train(g\_model, d\_model, gan\_model, dataset, latent\_dim)

CMRCET B. Tech (CSE) Page No 44

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 4**

**RESULTS AND DISCUSSION**

CMRCET B. Tech (CSE) Page No 45

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 4**

**RESULTS AND DISCUSSION**

**4.1 Performance metrics:**

The output from the project described above will be a generative model that can create novel images that resemble a set of training images. In other words, the goal of the project is to train a neural network to generate new images that share visual characteristics, patterns, and features with the images used for training.

Here's what the output will involve:

**1. Trained Generative Model:** The primary output of the project will be a trained neural network, specifically a generative model. This model will have learned the underlying distribution of the training images and will be capable of generating new images that follow a similar distribution.

**2. Generated Images:** Using the trained generative model, you will be able to generate new images that were not present in the original training dataset. These generated images will exhibit styles, textures, and structures similar to the images the model was trained on.

**3. Visual Quality and Realism:** The success of the project's output will be measured by the visual quality and realism of the generated images. The more realistic and indistinguishable from the training data these generated images are, the better the generative model has performed

**4. Diversity and Creativity:** A well-performing generative model will be able to produce diverse images that go beyond mere replication of the training examples. It should demonstrate creativity by generating variations and new instances that fit within the same style as the training images.

CMRCET B. Tech (CSE) Page No 46

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**5. Evaluation Metrics:** In addition to the generated images, the project might also involve metrics for evaluating the performance of the generative model. Common metrics include perceptual similarity measures, pixel-wise comparisons, and user evaluations.

**6. Documentation and Code:** The project output will likely include well-documented code that implements the generative model, training procedures, and image generation process. Documentation is essential for others (and yourself) to understand and reproduce the project's results.

**7. Visual Demonstrations:** Presentation of generated images side by side with real training images can provide a visual demonstration of the model's effectiveness in capturing the essence of the training data and generating similar content.

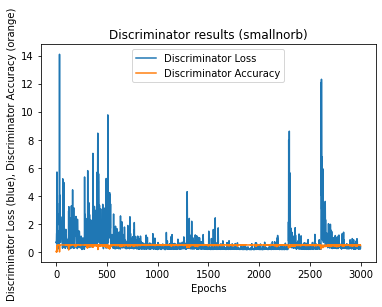
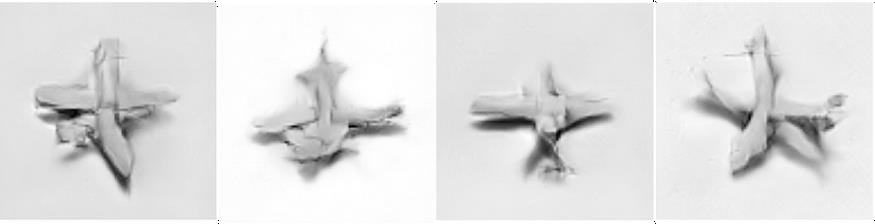


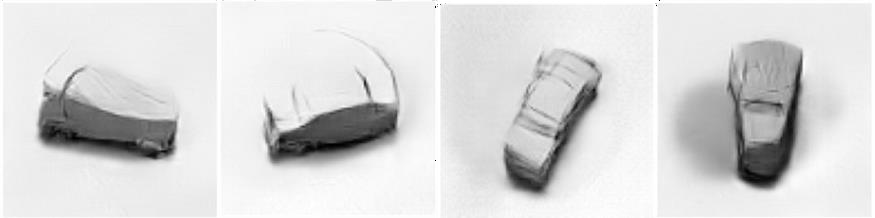
Figure 4.1.1: Graph of Performance Metrics

CMRCET B. Tech (CSE) Page No 47

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**









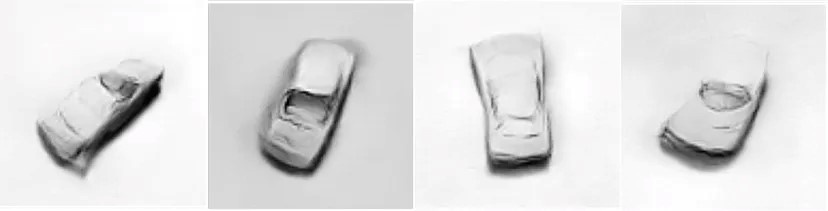


Figure 4.1.2: Generated Images by trained mode

CMRCET B. Tech (CSE) Page No 48

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

CHAPTER 5

**CONCLUSION**

CMRCET B. Tech (CSE) Page No 49

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**CHAPTER 5**

**CONCLUSION**

**Conclusion**

In conclusion, the "Generative Neural Networks for Novel Image Generation" project redefines the boundaries of neural network application by embarking on a journey of creativity and imagination. The resulting images encapsulate the innovative potential of generative models, amplifying their utility across domains that celebrate artistic innovation, data augmentation, and imaginative exploration.

This project not only expands the horizons of technology but also challenges us to think creatively and responsibly about the future of AI and its impact on society. As the field of AI continues to evolve, projects like this one will continue to inspire innovation, creativity, and responsible AI development

**5.1 Future Work**

* **Advanced GAN Architectures for Image Enhancement:** Develop and refine advanced Generative Adversarial Network (GAN) architectures to improve image fidelity and diversity. This involves exploring innovative approaches to training GANs that enhance the quality and variety of generated images.
* **Effective Style Transfer and Fusion Techniques:** Explore techniques for effective style transfer and fusion within GANs to encapsulate the essence of diverse training data. This includes investigating methods to seamlessly merge styles from different sources while maintaining visual coherence.
* **Privacy-Preserving GANs for Sensitive Data:** Investigate techniques to ensure privacy when using GANs for sensitive data. This involves developing methods to generate realistic data for secure information sharing without compromising confidentiality or revealing sensitive details.

CMRCET B. Tech (CSE) Page No 50

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**REFERENCES**

CMRCET B. Tech (CSE) Page No 51

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

## REFERENCES

1. StyleGAN: A Architecture -Based Generator for Generative Adversarial Networks by Tero Karras et al. (2019): This paper introduces StyleGAN, a powerful GAN architecture that allows for fine-grained control over the style and content of generated images.
2. GauGAN2: Efficiently Generating High-Quality Images from Semantic Layout by Junyu Lai et al. (2020): This paper introduces GauGAN2, a powerful GAN architecture that agrees for producing realistic images from simple semantic layouts.
3. GANs by Ian J. Goodfellow et al. (2014): This seminal paper introduces the subject of GANs and lays out the theoretical foundation for their operation.
4. Tim Salimans and colleagues' progressive Growth of GANs for enhanced quality, stability, and variation (2017): This paper proposes a progressive growth approach to training GANs, which allows for generating high-resolution images without sacrificing quality or stability.
5. FID Scores Are Easy to Manipulate by Martin Schiele and Shunyu Yao (2020): This paper highlights the limitations of commonly used metrics like FID (Fréchet Inception Distance) for evaluating the characteristics of produced images. It's important to consider this when comparing different GAN methods.
6. A Beginner's Guide to GANs by Jason Brownlee (Machine Learning Mastery): This article provides a good introduction to GANs, explaining the basic concepts and different architectures in an accessible way.
7. State-of-the-Art GAN Architectures Explained by Rameen Abdal (Towards Data Science): This piece delves into various cutting-edge GAN architectures, offering insights into their strengths and weaknesses.
8. Generating Images with StyleGAN2: A Practical Guide by Eliana Lorini (Hugging Face): This useful manual leads you through generating images with StyleGAN2, offering step-by-step instructions and resources.

CMRCET B. Tech (CSE) Page No 52

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**GITHUB LINK**

https://github.com/koushik1551/B-55

CMRCET B. Tech (CSE) Page No 53

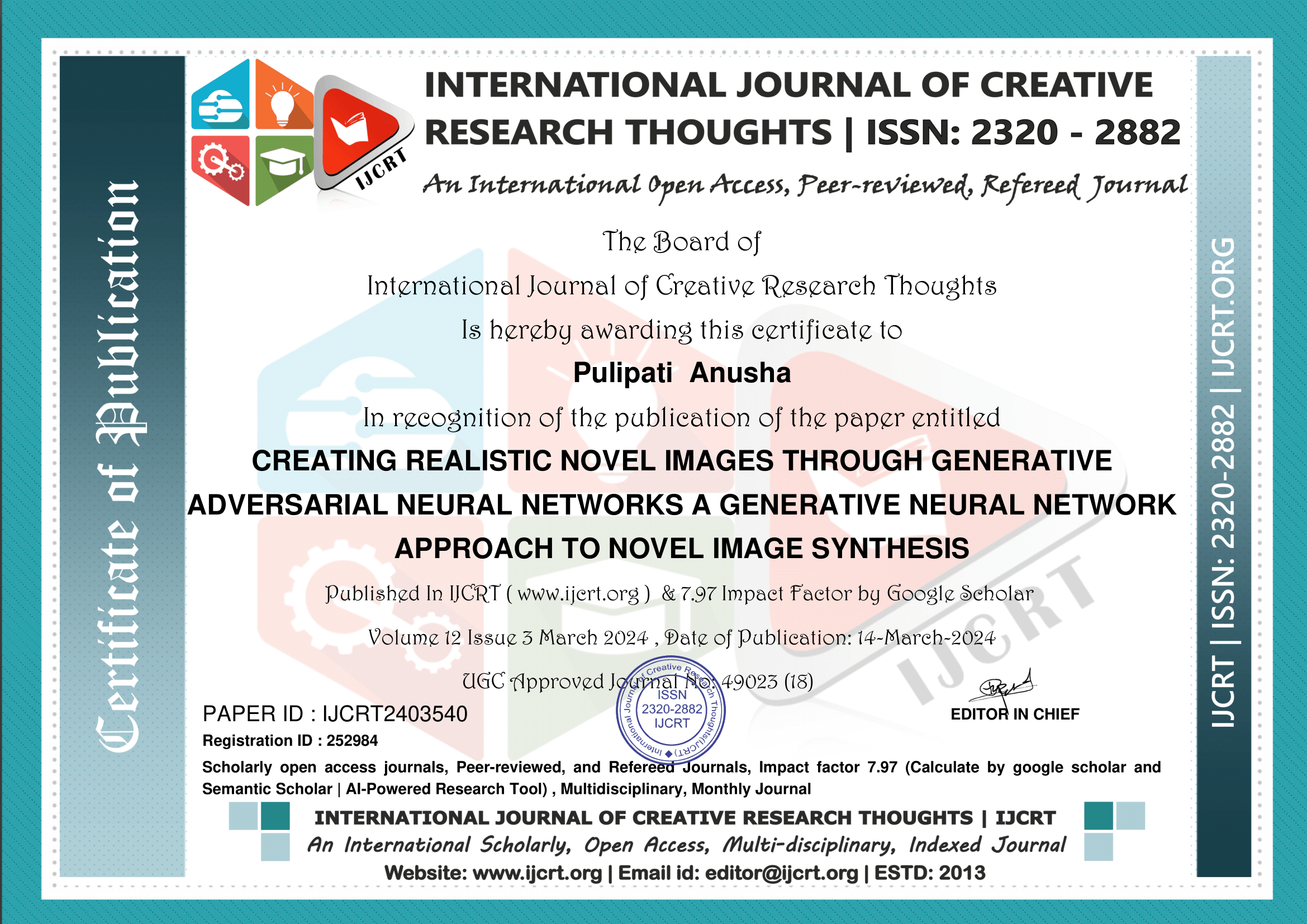
**Creating Realistic Novel Images through Generative Neural Networks (GAN)**

**PUBLICATION CERTIFICATES**



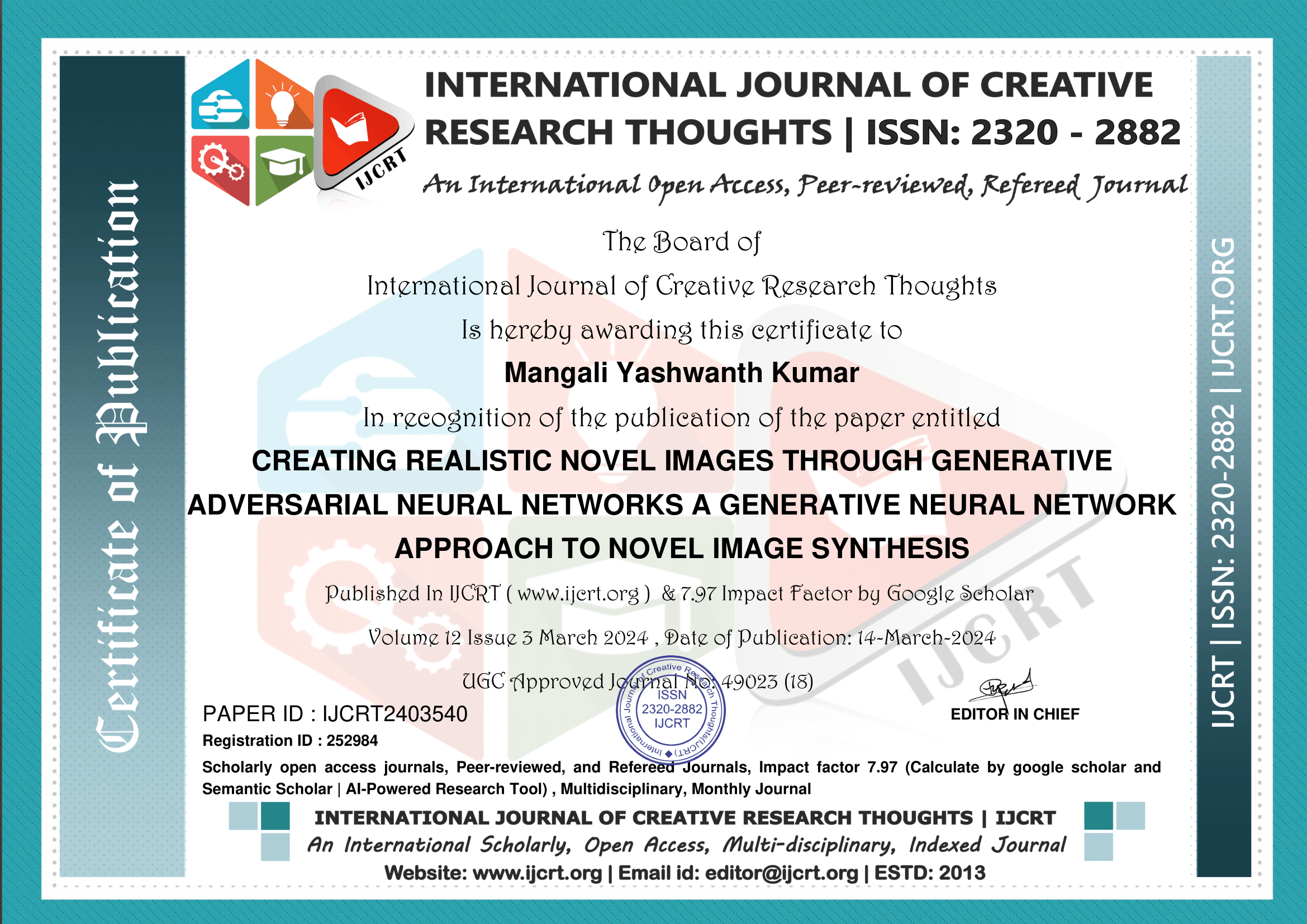
CMRCET B. Tech (CSE) Page No 54

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**



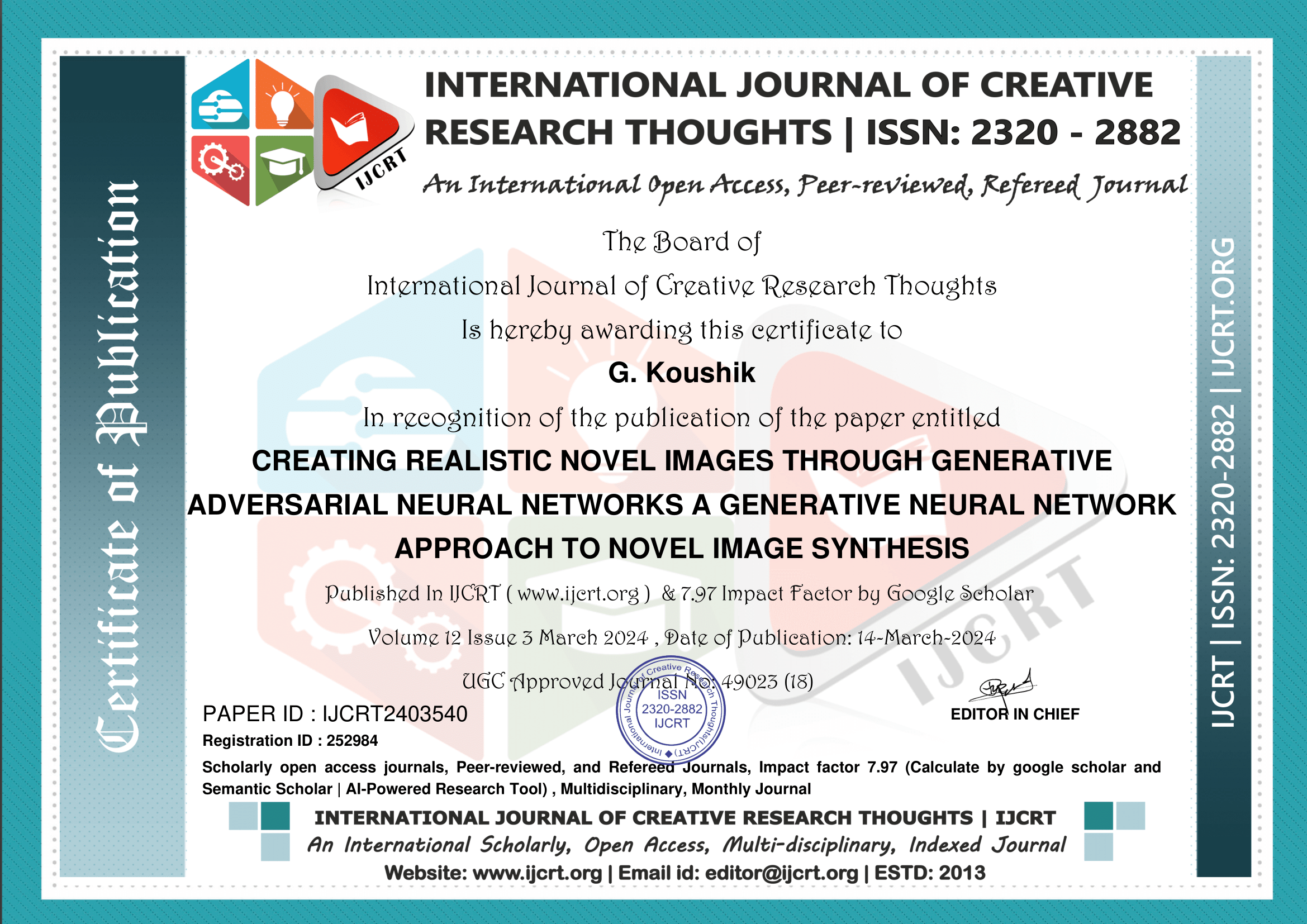
CMRCET B. Tech (CSE) Page No 55

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**



CMRCET B. Tech (CSE) Page No 56

**Creating Realistic Novel Images through Generative Neural Networks (GAN)**



CMRCET B. Tech (CSE) Page No 57